

# A New Approach: Component-Based Multi-Physics Coupling Through CCA-LISI

Fang Liu <sup>\*1</sup>, Masha Sosonkina <sup>\*2</sup>, and Randall Bramley <sup>+3</sup>

<sup>\*</sup> Scalable Computing Lab, USDOE Ames Laboratory  
Wilhelm Hall 329, Ames, IA, U.S.A 50010

<sup>+</sup> Computer Science Department, Indiana University-Bloomington  
Lindley Hall 215, 150 S. Woodlawn Ave. Bloomington, IN, U.S.A 47405

<sup>1</sup> fangliu@scl.ameslab.gov

<sup>2</sup> masha@scl.ameslab.gov <sup>3</sup> bramley@cs.indiana.edu

**Abstract.** A new problem in scientific computing is the merging of existing simulation models to create new, higher fidelity combined models. This has been a driving force in climate modeling for nearly a decade now, and fusion energy, space weather modeling are starting to integrate different sub-physics into a single model. Through component-based software engineering, an interface supporting this coupling process provides a way to invoke the sub-model through the common interface which the top model uses, then a coupled model turns into a higher level model. In addition to allowing applications to switch among linear solvers, a linear solver interface is also needed for the model coupling. A linear solver interface helps in creating solvers for the integrated multi-physics simulation that combines separate codes, and can use each code's native and specialized solver for the sub-problem corresponding to each physics sub-model. This paper presents a new approach on coupling multi-physics codes in terms of coupled solver, and shows the successful proof for coupled simulation through the implicit solve.

**Key words:** Parallel model coupling; Component architecture and interface; Sparse matrix computation

## 1 Introduction

Modeling physical phenomena with scientific computing is an interdisciplinary effort. Many problems in science and engineering are best simulated as a set of mutually interacting models; practical physical systems are often mathematically modeled by complicated Partial Differential Equations (PDEs). Many real-world systems involve a complex of multiple physical components. Scientists in many fields are becoming increasingly interested in coupling models together in order to advance their understanding. For example, the Sun-Earth system [29] presents a complex natural system of many different interconnecting elements: the solar wind transfers significant mass, momentum and energy to the magnetosphere, ionosphere, and upper atmosphere, and dramatically affects the physical processes in each of these physical domains. The Community Climate System Model (CCSM) [7] for global climate models comprises interdependent models

that simulate the Earth’s atmosphere, ocean, cryosphere, and biosphere. These systems interact by exchanging energy, momentum, moisture, chemical fluxes, etc. For example, atmosphere provides to the Earth’s surface downward radiative fluxes, momentum fluxes in the form of wind stress, and fresh water flux in the form of precipitation. Fusion energy simulation is now integrating codes that model different physics of a fusion reactor [11, 20]. Most recently the Center for Simulation of RF Wave Interactions with Magnetohydrodynamics (CSWIM) [6] works on coupling existing codes to model the interaction between high power radio frequency (RF) electromagnetic waves, and magnetohydrodynamics (MHD) aspects of the burning plasma.

In various scientific simulation domains, a stand-alone model can run with simplified assumptions on the interaction of the particular domain with the rest of the system. Merging the existing simulation models to create a new higher fidelity combined model is a newly emerging theme in the scientific computing community. Data exchange based model coupling such as Model Coupling Toolkit (MCT) [22] and Earth System Modeling Framework (ESMF) [31] proves that models can be coupled through exchanging the boundary or initial condition between models. Models are generally mesh-based, time-stepping models and may be based upon highly complex or nonlinear algorithms and numerical schemes. The process of model coupling can occur on three different time scales: tightly coupled models operate at the fastest time scale where data are exchanged every time step; for slowly varying physics, the coupling needs only be done every few time steps; the slowest coupling operates once per simulation. In this paper, we won’t get into the details of the above complexities, and mainly focus on the framework support in which models are to be embedded in order to form part of a coupled application.

Software component model for scientific computing may help in coupling the multiple physics codes, such that each code presents the clear interface for the center controller - **Coupler**. Our previous research work [23, 24, 27] designed a common interface LInear Solver Interface (LISI) which spans multiple high-performance computing (HPC) solver packages. Within Common Component Architecture (CCA) framework CCAFFEINE [8], each package can be encapsulated into a component providing standard interface so that easy switching of solver packages is achieved. In this paper, each sub-physics maps into an individual solver, a coupled physics becomes a coupled solver. It demonstrates the idea for coupling multi-physics codes through LISI interface by using implicit solve method. We first investigate some efforts on model coupling; then present the requirements for solver coupling; after analyzing the design in details, we provides a CCA coupling approach; finally we give a test case to validate the solver coupling idea.

## 2 Related Work

Some of coupling efforts have been done in the scientific computing community over the past 10 years. They have been deployed within their science domain [22, 17, 31, 29, 26, 19]. None of them supports the wide variety of discretization schemes and numerical techniques of existing discretization schemes, and combining codes from different frameworks is still hard.

Tools such as MCT [22] that tries to provide a common utility to couple the climate models are mainly focusing on the data exchange between the models. The ESMF [31] defines an architecture for composing complex, coupled modeling systems. The complicated applications are broken up into smaller pieces (components). Components are assembled together to create an application and the different implementations of component can be used in a plug-N-play (plug-N-play) fashion. The Space Weather Modeling Framework (SWMF) [29] aims at providing a flexible and extensible software architecture for multi-component physics-based space weather simulations. The SWMF uses layer architecture which is similar to the ESMF. A solid rocket motor simulation [19] requires consideration of multiple physical components, such as fluid dynamics, solid mechanics and combustion. Roccom provides an object-oriented software integration framework for inter-module data exchange and function invocation in parallel multi-physics simulations. The undergoing CSWIM [6] project is focusing on the interaction between high power radio frequency (RF) electromagnetic waves, and magnetohydrodynamics (MHD) aspects of burning plasma. It uses the batch management system and even system to couple the codes.

The above work demonstrates the successful usage in their perspective domains, but none of these frameworks has attracted a large user base or been widely adopted outside their field of application due to its lack of generality. CCA [14] has been started to address this problem, applying component principles at the level of whole applications, so that parallel applications can run both stand-alone and with other applications within the framework.

### 3 Solver Coupling Requirements

Coupled modeling is increasingly necessary to make progress in understanding the science of complex physical phenomena. The interaction in a combined simulation needs to be addressed by mathematical and physical aspects of simulation. Combined problem can be solved by forcing consistent solutions on the interfaces. The combination is not trivial because the constituent applications come with their own meshes, discretizations and internal data structures, especially in HPC, the data decomposition may be different.

Component models have been introduced to the module-coupling community recently, and already found themselves well suited for the requirements. The most promising approach is to define a standard set of interface functions that every physics component must provide. The interface describes the list of inputs and outputs, and exists independent of the implementation. Such common interfaces are present in some of the frameworks mentioned in section 2. Each sub-model needs to provide standard interface to the framework while the framework provides the data exchange interface for each sub-model to use.

In this section we analyze some of the challenges pertinent to the treatment of combined physics models as a coupled solver in a multi-physics simulation. The invocation of sub-model is through CCA LISI interface [23]. The target models include those that have

- an extensive software base of existing codes which were not designed to interoperate with other codes.
- an investment in the sub-model software which is too expensive to duplicate because of various reasons such as continuing active use, development, and evolution. In other words, it is not sufficient to use an earlier release or frozen version.
- an “awareness” of the other sub-physics, typically by a greatly simplified treatment of their contributions. For example, they assume that a boundary condition is not time-variant, or that some averaging is sufficient to reflect contributions from other sources, or that the constituent physics are not coupled.
- a vital need for inter-model interfaces, which requires active collaboration between application and computer scientists. The definition of those interfaces is ideally a community effort with broad intellectual support.
- parallelism requirement in the computations.

Through the participated models’ feature, we may abstract the major requirements for the solver coupling:

- supporting multi-language legacy codes,
- providing data exchange sub-model running on the different size of processes (called MxN problem),
- requiring minimal changes for each physic model,
- allowing individual code evolvement,
- use of model’s native and possibly highly specialized solvers.

In the following subsections, we will give detailed explanations on above requirements.

### 3.1 Multi-language support

The large scale simulation codes always involve the modules developed by the different teams from different institutes. The code may be in different program languages, parallelism paradigms or platform dependability. The CCA [14] provides HPC language support, in particular, support for FORTRAN77, FORTRAN90, C, C++, Java, etc. This is achieved through Babel [16], a tool relying on the Scientific Interface Definition Language (SIDL) [21] to express software interfaces in a language neutral way. Babel compiler can generate the appropriate glue code stubs and skeletons based on SIDL to facilitate language interoperability. In turn, CCA supports the application to run with components of different languages.

### 3.2 Data exchange

In a coupled model, the data exchanged by two components  $A_i$  and  $A_j$  reside on their overlap domain  $\Omega_{ij}$ , and in principle each component will have its own discretization of  $\Omega_{ij}$ . And two components may run on different numbers of processors, so that the data partitioning on the overlap domain may be different. This is so called M-by-N problem in the scientific coupling [12] community. In our design, we treat sub-models as in an algebraic way, such that we assume that some other components already did the data interpolation and M-by-N data exchange before the coupling interface is called. In our

current research, we only target on the case when the coupled models run on the same number of processors, but in the future, when considering the other cases, we will take a look at how to integrate the M-by-N component with our coupling framework.

### 3.3 Minimal code changes

Existing software integration frameworks typically require large manual rewrites of existing codes, or specific tailoring of codes written to be used in the framework. The resulting special purpose code is not usable outside of the framework. The easy reuse of legacy software is one of effort of CCA [14], it minimizes the effort required to incorporate existing software into CCA environment. A thin layer of the component wrapper is added to the legacy code, with well defined interface the legacy code becomes a ready to use component. Since the target models are too expensive to rewrite, the above solution makes the minimal code change possible.

### 3.4 Individual code evolvement

The coupled model should allow each sub-model keep growing as long as the interface it provides is unchanged. In a coupled solver, each sub-model provides a LISI interface, only the adapter needs to be updated once the sub-model grows. The Component Based Software Engineering (CBSE) allows individual sub-model evolving without disturbing other participating sub-models.

### 3.5 Use of the native solvers

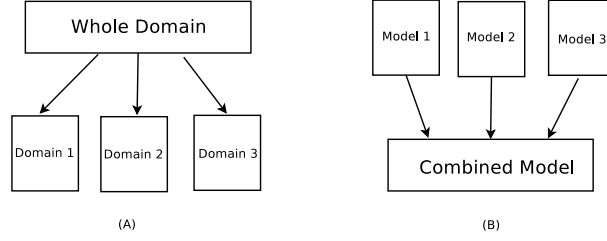
In a PDEs simulation, sometimes the general solver cannot fully solve the discretized problem due to complex geometry, and mesh shape, and discretized form of PDEs. The resulting linear system may have a very irregular sparsity pattern, large condition number, etc., which makes iterative method such as *GMRES*, *BICGSTAB* impossible to solve it. In this situation, it is hard to use the available solver packages. These simulations usually write their own solvers, in some way the solver is embodied with the simulation code. If this model is coupled with other models, the exposed LISI interface can make the model a special solver component.

While using CCA and LISI to define the coupled solver, the above requirements can be satisfied. And when each sub-physics is treated as a solver, the coupled solver idea can be used in multi-physics coupling.

## 4 Design of Coupled Solver

In order to analyze the coupled solver, some mathematics background needs to be demonstrated first. Domain decomposition method received a strong revival of interest in the end of 80s and early 90s due to its potential in parallel computing [15]. It is a class of techniques for the solution of PDEs on a domain by solving smaller problems on subdomains. They are particularly useful for solving problems on irregular domains and on parallel computers. The key ingredient is the system of equations governing the

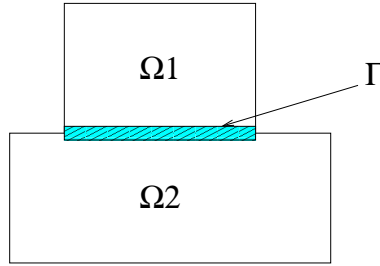
variables on the interface between the subdomains. The domain decomposition method idea can be traced back to Schwarz’s alternating procedure, in which existence of solutions to boundary value problems are proved by an iteration involving solutions on overlapping subdomains. This idea is also widely used in many fields of scientific computing. Figure 1 shows these two ideas, and our current research borrows the domain decomposition idea in a reverse manner.



**Fig. 1.** (A) Domain decomposition used in parallel computing about 20 years ago; (B) Emerging modeling coupling from individual sub-model

#### 4.1 Model Description

To simplify the illustration, we consider two-domain problem in Figure 2. There are two overlapping subdomains  $\Omega_1$  and  $\Omega_2$ , the interface between two domains is named  $\Gamma$ .



**Fig. 2.** Coupled domain

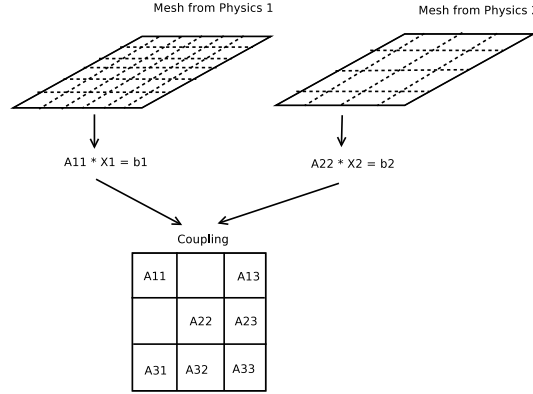
When the domain decomposition method idea is applied to multi-physics coupling, the domain  $\Omega_1$  and  $\Omega_2$  can be treated as the different physics simulations, which solve the different PDEs, for example. There are two cases for the problem to be solved

1. when there is no interaction between physics, the discretized system is solved as:

$$\begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \quad (1)$$

Here  $A_{11}$  and  $A_{22}$  are discretized form of each physics on its own domain,  $x_1$  and  $x_2$  are solution vectors for each physics,  $b_1$  and  $b_2$  are the right hand side vectors for each physics simulation.

2. When thinking about coupling two physics codes, a combined linear system is created in Figure 3. Besides  $A_{11}$  and  $A_{22}$  for each physics, a few new matrices are



**Fig. 3.** Multi-Physics Coupling Diagram

introduced through the interaction between two physics:

- $A_{33}$  represents the linear system for interface nodes, it is based on governing PDEs used on the interface nodes, it may have its own discretization scheme used or use one of two coupled physics discretization scheme, which is the decision made by two physics simulation codes. Some data mapping needs to be done on these interface nodes. For example, the grid points of one simulation may correspond to the mesh of the coupled simulation. Usually data interpolation and MxN component need to solve this problem as we discuss in section 3.
- $A_{13}$  and  $A_{13}^T$  are the coupling matrices between physics on  $\Omega_1$  and interface nodes, generally the interactions between them are symmetric from each side of point of view, so that the transpose of  $A_{13}$  represents the coupling from interface nodes to the nodes on the physics domain  $\Omega_1$ .
- $A_{23}$  and  $A_{23}^T$  are the coupling matrices between physics  $\Omega_2$  and interface nodes.
- $x_3$  and  $b_3$  are the solution vector and right hand side vector for the interface nodes.

$$\begin{bmatrix} A_{11} & 0 & A_{13} \\ 0 & A_{22} & A_{23} \\ A_{13}^T & A_{23}^T & A_{33} \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad (2)$$

Through comparing non-coupling Equation 1 and coupling Equation 2, the system of linear equations becomes more complex to solve, especially when each sub-physics simulation runs in parallel.

## 4.2 Solver Methods

In this section, we are investigating two solver methods on solving Equation 2. Although both methods can be used, one is chosen over the other due to its guaranty on solver's convergence.

One of the most popular method on solving Equation 2 is the alternating Schwarz method [25]. The original alternating procedure described by Schwarz in 1870 consisted of three parts:

1. Alternating between two overlapping domains,
2. Solving the Dirichlet problem on one domain at each iteration,
3. Taking boundary conditions based on the most recent solution obtained from the other domain.

This procedure is called the Multiplicative Schwarz procedure [13, 18]. This algorithm is therefore quite sequential, since each sub-domain updates its unknowns based on the other domain's previous step solution. It may not be the best candidate for parallel model coupling. The additive variant of the Schwarz procedure is more suitable for parallel processing, because the subdomains are not updated until a whole cycle of updates through all domains are completed [25], and the subdomains can be solved separately and at the same time.

When thinking coupled physics problem in terms of coupled solver, alternating Schwarz method is one of the options. But this method is slow and sometimes not convergent. There are two key assumptions associated with classical Schwarz method, and these assumptions are not verifiable from linear algebra arguments alone, see [25] (chapter 13.3). Given a linear system, it is unlikely that one can analytically establish that these assumptions are satisfied.

Another way to solve this problem is the Schur complement method [30]. The Schur complement arises naturally in solving (2) by using Block-Gaussian Elimination:

$$G = A_{33} - A_{13}^T * A_{11}^{-1} * A_{13} - A_{23}^T * A_{22}^{-1} * A_{23},$$

where  $G$  is the Schur complement system for solving the interface nodes, and  $G$  is much smaller than  $A_{22}$  and  $A_{33}$ . The solution of interface nodes then can be used to compute the solution for each physics subdomain. As we mentioned in section 1, we introduce the concept of **Coupler** to capture the interaction between two coupled physics codes. Assume that physics codes know nothing about the coupling matrix  $A_{13}$  and  $A_{23}$  which are only known to the coupler. **Coupler** is an additional program doing extra work alongside two stand-alone physics simulations.

## 4.3 Coupled Solver Algorithm

In our approach, we chose the Schur complement method for our coupled solver, the reasons are at two folds: (1) Alternating Schwartz method is slow and sometimes not convergent. It can only couple the systems via their boundary conditions; (2) Schur complement method converges in fewer iterations for any coupled system and explicitly account for complex coupling interactions.

These solution processes can be done with Krylov subspace methods such as GMRES and BICGSTAB [10], which requires repeated matrix-vector multiplications. The algorithm 1 details the solution process.

---

**Algorithm 1** Coupling Algorithm through Schur-complement Method
 

---

Initialize:

- subdomain  $\Omega_1$  computes  $b_1$  and  $\omega_1 = A_{11}^{-1} * b_1$ , return  $\omega_1$  to coupler;
- subdomain  $\Omega_2$  computes  $b_2$  and  $\omega_2 = A_{22}^{-1} * b_2$ , return  $\omega_2$  to coupler;
- coupler computes  $b_3 = b_3 - A_{13}^T * A_{11}^{-1} * b_1 - A_{23}^T * A_{22}^{-1} * b_2$

Step 1: Solve  $G * x_3 = b_3$ : During each iteration, a matrix-vector product is conducted, ( $x_{3p}$  represents the solution vector of  $x_3$  from the previous iteration), and steps include:

1. coupler computes  $x_3 = A_{33} * x_{3p}$
2. coupler computes  $d_1 = A_{13} * x_{3p}$
3. subdomain  $\Omega_1$  computes  $\omega_1 = A_{11}^{-1} * d_1$
4. coupler computes  $x_3 = x_3 - A_{13}^T * \omega_1$
5. coupler computes  $d_2 = A_{23} * x_{3p}$
6. subdomain  $\Omega_2$  computes  $\omega_2 = A_{22}^{-1} * d_2$
7. coupler computes  $x_3 = x_3 - A_{23}^T * \omega_2$

Step 2: Compute

- $x_1 = \omega_1 - A_{11}^{-1} * A_{13} * x_3$
  - $x_2 = \omega_2 - A_{22}^{-1} * A_{23} * x_3$
- 

Note that during each iteration, when Schur complement matrix is used, five matrix-vector products are called, and two linear solvers are deployed. The algorithm only requires the solution of problems with  $A_{11}$  and  $A_{22}$ , which corresponds to solving independent problems on the subdomains. This means that coupled solver can leave the subdomain solve unchanged while adding new functionality only in the coupler.

## 5 CCA Coupling Approach

Component-based software design combines object-oriented design with the powerful features of well-defined interfaces, programming language interoperability and dynamic composability [28]. While component-based design was initially motivated by the needs of business application developers, it also offers enormous potential benefits to the computational science community. The CCA [14] is a software component model that specially addresses HPC applications. In CCA, the software unit is treated as a component, each component connects to another component through a collection of public interface, or ports [1]. CCA employs a provides/uses paradigm in which a component provides a set of interfaces that other components can use.

In our previous research work, LISI [23, 24, 27] is an effort within CCA Forum to identify the common requirements among widely available HPC sparse linear solver

packages, and abstract a common Application Programming Interface (API) that spans them. LISI is designed to facilitate the run time plug-N-play from multiple HPC solver packages. The generic solver interface *lisi.SparseSolver* is deployed as a CCA port which can be implemented by a solver component and used by another component. The auxiliary interface when a matrix free solver is used is also presented as *lisi.MatrixFree* interface. The component providing this interface will provide a matrix vector product functionality, and the component using this interface will be a solver component of solving a linear system in a matrix free manner. In order to support constructing a coupled solver for the multi-physics coupling simulation, the *lisi.BaseSolver* is abstracted from *lisi.SparseSolver* to support the matrix free solver. This interface only requires *uses* port to provide right hand side vector and get the solution back, and the block row partitioning is assumed in this interface.

\_\_\_\_\_LISI BaseSolver Interface\_\_\_\_\_

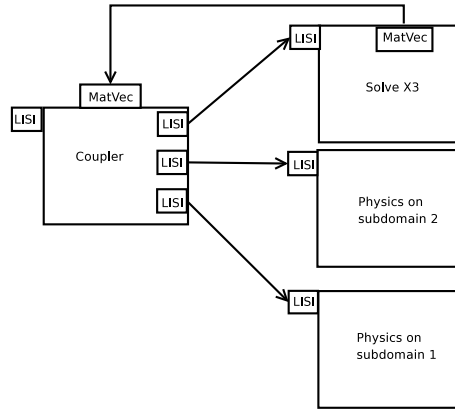
```
package lisi version 0.2
{
  interface BaseSolver extends gov.cca.Port
  {
    int initialize(in long comm);
    int setStartRow(in int startrow);
    int getStartRow();
    int setLocalRows(in int rows);
    int getLocalRows();
    int setupRHS(
      in rarray<double,1> RightHandSide(NumLocalRow),
      in int NumLocalRow);
    int solve(
      inout rarray<double,1> Solution(NumLocalRow),
      inout rarray<double,1> Status(StatusLength),
      in int NumLocalRow,
      in int StatusLength);
  }
}
```

## 5.1 Design Architecture

In section 4, we have analyzed the coupled solver in details from its mathematics aspect. From the algorithm we present in previous section, there must be at least three components for our CCA design, one for each subdomain physics simulation and one for the coupler. Since the coupler is also going to solve the Schur complement system for  $x_3$ , we introduce another solver component to solve the Schur complement system in a matrix free manner so that the Schur complement system does not have to be formed explicitly.

Next question we need to answer is what information each component should hold to best simulate the real world coupling situation? In our design we treat the coupled physics simulation as a coupled solver, for two physics subdomains, they should hold their own linear systems  $A_{11}$  and  $A_{22}$ , and right hand side vectors  $b_1$  and  $b_2$ . Since

physics simulation maintains its own application data, it runs as a stand alone application but now provides *lisi.BaseSolver* port. All the coupling information should retain within **Coupler** component, the information includes the coupling matrix between physics subdomain 1 with the interface nodes -  $A_{13}$ , and the coupling matrix between physics subdomain 2 with the interface nodes -  $A_{23}$ , and the discretization on the interface nodes with its own governing function -  $A_{33}$ . In the real world coupling simulation, these three matrices must be designed by application domain expert in order to construct a meaningful combined simulation.



**Fig. 4.** Coupling With CCA Components

Figure 4 shows the design architecture for the coupled physics simulation, the component diagram follows the CCA uses-provides design pattern, the arrow means the calling direction.

- **Coupler** component has two provides ports: one is *lisi.BaseSolver* to provide the functionality to the outside application who may treat the coupled simulation as a solver; one is *Matvec* port which provides the matrix vector product for the Schur complement system when solving  $x_3$ . This component also has three uses ports, and all of them are *lisi.BaseSolver* ports. These ports are used to get the solution back from two physics simulation components and one solver component for  $x_3$ .
- **Physics** components on subdomain 1 and subdomain 2, they are basically converted from real world simulation codes to components through a provides port *lisi.BaseSolver*. Usually a thin layer is implemented on top of the legacy codes. These two components are called once during each iteration in the coupling algorithm as described in Section 4.
- **Solver** component for  $x_3$ . It represents Step 1 in algorithm from Section 4. Since this component has to solve a Schur complement system in a matrix free fashion, it has a uses port of *MatVec* along with its provides port *lisi.BaseSolver*.

## 5.2 Implementation

This design makes **Coupler** as a central hub for the coupling, and **Physics** components and  $x_3$ -**Solver** component should run simultaneously to exploit concurrence to the most in the simulation overall. CCA is communication transparent specification, which makes it lightweight and simple to use [9]. In our design, we choose in the most intuitive way among several MPI constructions already known by application developers, such as MPI communicator groups. Components in the CCA coupling run in a single MPI instance (started by a single *mpirun* command). The MPI communicator world needs to be divided into several subgroups as follows:

- `a1_group` and `a2_group` are MPI communicator groups for Physics component on subdomain 1 and subdomain 2, respectively. Depending on how large each simulation is, these two groups differ in size. And they shouldn't overlap to each other, since two simulations need to run simultaneously.
- `a3_group` is for both Coupler component and  $x_3$ -Solver component. Since every time Coupler requires from  $x_3$ -Solver, it has to wait for the solution return in a blocking call manner. It is not crucial to have concurrency between these two components. Since the  $A_{33}$  is much smaller compared to  $A_{11}$  and  $A_{22}$ , the processes assigned to Coupler and  $x_3$ -Solver might have number fewer than the Physics component. And `a3_group` shouldn't share processes with either `a1_group` or `a2_group`.
- `a13_group` is the augmented MPI communicator group for both `a1_group` and `a3_group`, and `a23_group` is the augmented MPI communicator group for both `a2_group` and `a3_group`. These communicator groups are used for data passing between Coupler component and Physics components.

During initialization phase, two **Physics** components load their linear system  $A_{11}$  and  $A_{22}$ , and the right hand side vectors  $b_1$  and  $b_2$ , act as a real application. **Coupler** component loads the coupling information  $A_{13}$ ,  $A_{23}$ ,  $A_{33}$  and  $b_3$ . Since all the components may run on different number of processors, data is partitioned into the different number of chunks. In our design, we tried to avoid **MxN** problem [12], so we make the physics components running on the same number of processors, and the coupler running on a single processor. But in the real coupling problem, when adding the **MxN** component, both coupler and physics components can run on any number of processors. In order to get the solution from subdomain 1 back to the coupler, solution is produced within MPI communicator group `a1_group`, but collected to coupler's processor through communicator group `a13_group`. The similar is true for the solution on subdomain 2.

During the solve phase, **Solver** component calls back through *MatVec* port during each of its iteration for solution of  $x_3$ . During each iteration step, solutions of  $A_{11}$  and  $A_{22}$  are needed from physics subdomain 1 or subdomain 2. The similar steps are applied as in initialization phase. One different thing is that now the right hand side vector needs to be sent from the coupler to each physics component, MPI scatter call is used within each augmented communicator groups. Also some synchronization should be done for signaling these solving processes such as a Boolean variable indicating the process starts. Basically physics components are waiting for the signal of *MatVec* from Coupler. Only if the signal is received, they participate the collective calls on *MPI\_Scatterv* and *MPI\_Gatherv*.

During the final computing phase, the newly computed solution from Schur complement system on  $x_3$  is used to compute the solution vector for each physics subdomain as indicated in section 4.3. Now the coupling cycle is done, two physics problems are solved under the condition that they interact each other on the overlapped domain.

### 5.3 Combining Multiple Packages

In our implementation, we also want to demonstrate that multiple solver packages can be linked together to compose into a new higher fidelity solver. Since in our previous research [23], we have converted Trilinos AZTECOO [5], PETSc [3] and SuperLU [4] into CCA components, they have provided *lisi.SparseSolver* port to other components to use. Now we need to reimplement those components with *lisi.BaseSolver* interface, and another new package is added - High Performance Preconditioners (HYPRE) [2]. HYPRE is a multigrid solver, and Hype's BOOMERAMG is a algebraic multigrid solver. Our goal is to demonstrate the interface usage on coupling code, which allows each code to use its native (specialized) solver. We choose the AZTECOO solver for our physics simulation on subdomain 1, used the BOOMERAMG solver for our physics simulation on subdomain 2, and the PETSC solver as Schur complement solver for  $x_3$ . In this way, three widely used solver packages are deployed together in one application within a component based multi-physics simulation. Not only the idea of combining the multiple solver packages is validated, but also multi-physics coupling can be resolved.

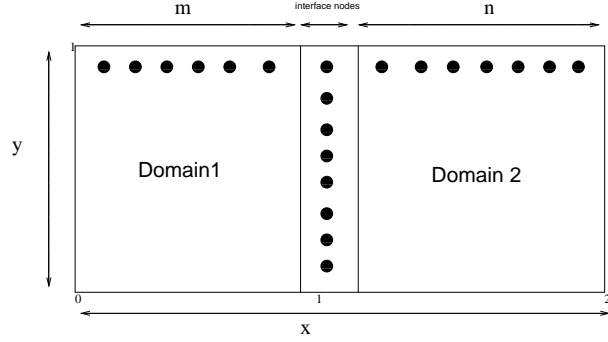
## 6 A Test Case

In order to demonstrate the multi-physics coupling through the LISI interface, we build a prototype test problem which may arise in a typical real-world application scenarios. There are two domains both model the following 2-dimensional PDEs,

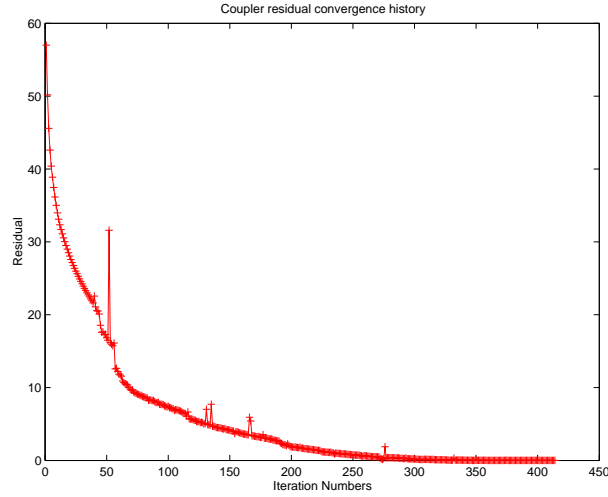
$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f \quad (3)$$

with Dirichlet boundary conditions, discretized with five-point centered finite-difference scheme on  $n_x \times n_y$  grid. Where  $f = (2.0 - 6.0 * x - x^2) * \sin(y)$  and boundary condition  $b = x * x * \sin(y)$ . Figure 5 shows our test problem, domain 1 sits on the left side with domain boundary  $[0, 1] \times [0, 1]$ , domain 2 sits on the right side with  $[1, 2] \times [0, 1]$ . The interface nodes align vertically at  $x = 1$  between two domains. And the interface domain is discretized with one dimensional PDEs. 400 discretized nodes are used along  $x$ - and  $y$ - direction, so the linear system order is about 160000. The same discretization is used for both domains. The test runs on the Linux cluster *Odin* in the Computer Science Department at Indiana University. *Odin* has 128 nodes, each with two dual core AMD Opteron 2.0 Ghz processor and 4GB RAM on each computing node.

Physics subdomain 1 runs Trilinos AZTECOO solver with maximum iteration number 500 and tolerance of  $1.0e^{-6}$ , the solver method BICGSTAB and preconditioner method Jacobi are used. Physics subdomain 2 runs HYPRE's BOOMERAMG solver with maximum iteration number 30, and tolerance of  $1.0e^{-6}$ , all other parameters are set to default. Schur complement system solver for  $x_3$  runs Portable, Extensible Toolkit

**Fig. 5.** Test problem setup

for Scientific Computation (PETSc)'s BICGSTAB method with maximum iteration number 500 and tolerance of  $1.0e^{-6}$ , since it uses PETSc matrix-free method, there is no preconditioner chosen. The test runs on 9 processes total, components for physics 1 and 2 each run on 4 processes, and coupler runs on 1 process. The test repeats for 10 times, and result is average of the runs. Figure 6 shows the history of residual from solving the Schur complement system on  $x_3$ , the residual decreases as the iteration number increases, and it converges at the iteration number 415. This result demonstrates that coupled system is successfully solved. Sub-physics on domain 1 solves in 286 iterations with Trilinos solver while sub-physics on domain 2 solves in 5 iterations with HYPRE solver.

**Fig. 6.** Test result

## 7 Conclusion

In this paper, we demonstrate a new way of coupling multi-physics codes through CCA-LISI. This is the first model coupling approach within CCA, we generalize the model coupling idea through CBSE and treat coupling models as an abstracted solver. We outline a flexible approach to creating coupled model by introducing a new concept - coupled solver. When each sub-physics is treated as a sub-solve from the coupled physics, an implicit solve between multiple sub-physics can become a coupled solver. With CCA [14] technology, each sub-physics is encapsulated as a component with standard interface exposed to other components. It makes coupling easier through introducing extra component - **Coupler**. While each sub-physics model still runs separately, they all talk to **Coupler** to exchange the coupling information. The paper analyzes the requirements for a coupled solver, suggests two coupling algorithm and compares them. It also presents a coupling algorithm design along with its detailed implementation through CCA component framework; Although the tests were conducted with rather simple physics models, they nevertheless clearly show the coupled solver in action, and thus validate the proposed coupling of physics models under certain assumptions of data representation in algebraic form. As a future work, a generic way to represent the coupling information will need to be considered. For example, data interpolation needs to be done by other components when sub-models run on different numbers of processes. Being sufficiently general, the LISI interface, successfully used in this work, may be extended to incorporate new representations of the coupling information.

## References

1. CCA specification. <http://www.cca-forum.org/specification>, April 2008.
2. HyPre: Scalable Linear Solvers : high performance preconditioners. [http://www.llnl.gov/CASC/linear\\_solvers/](http://www.llnl.gov/CASC/linear_solvers/), April 2008.
3. PETSc: Portable Extensible Toolkit for Scientific Computation. <http://www-unix.mcs.anl.gov/petsc/petsc-as/>, April 2008.
4. SuperLU: Direct Solver Package of large, sparse, nonsymmetric systems of linear equations. <http://crd.lbl.gov/~xiaoye/SuperLU/>, April 2008.
5. The Trilinos Project. <http://software.sandia.gov/trilinos>, April 2008.
6. Center for simulation of rf wave interactions with magnetohydrodynamics. <http://cswim.org>, March 2009.
7. Community climate system model. <http://www.cesm.ucar.edu/>, March 2009.
8. B. A. Allan and R. Armstrong. Caffeine Framework: Composing and Debugging Applications Iteratively and Running them Statically. Compframe 2005 workshop, June 2005.
9. B. A. Allan, R. C. Armstrong, A. P. Wolfe, J. Ray, D. E. Bernholdt, and J. A. Kohl. The cca core specification in a distributed memory spmd framework. *Concurrency and Computation: Practice and Experience*, 14(5):323–345, 2002.
10. R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. V. der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994.
11. D. Batchelor. Integrated Simulation of Fusion Plasmas. *Physics Today*, 58:35–40, 2005.
12. F. Bertrand, Y. Yuan, K. Chiu, and R. Bramley. An approach to parallel MxN communication. In *Proceedings of the Los Alamos Computer Science Institute (LACSI) Symposium*, Santa Fe, NM, October 2003.

13. P. E. Bjørstad. Multiplicative and additive schwarz methods: Convergence in the two-domain case. In *Domain Decomposition Methods*, T. Chan, R. Glowinski, J. Periaux, and O. Widlund, eds., pages 147–159, Philadelphia, PA, 1989. SIAM.
14. CCA-Forum. The DOE Common Component Architecture project. <http://www.cca-forum.org/>, April 2008.
15. T. F. Chan and D. C. Resasco. A Framework for the Analysis and Construction of Domain Decomposition Preconditioners. In *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*.
16. T. Dahlgren, T. Epperly, G. Kumfert, and J. Leek. *Babel User's Guide*. CASC, Lawrence Livermore National Laboratory, Livermore, CA, babel-0.11.0 edition, 2005.
17. L. A. Drummond, J. Demmel, C. R. Mechoso, H. Robinson, K. Sklower, and J. A. Spahr. A data broker for distributed computing environments. In *ICCS '01: Proceedings of the International Conference on Computational Sciences-Part I*, pages 31–40, London, UK, 2001. Springer-Verlag.
18. M. Dryja. An additive schwarz algorithm for two- and three- dimensional finite element elliptic problems. In *Domain Decomposition Methods*, T. Chan, R. Glowinski, J. Periaux, and O. Widlund, eds., pages 168–172, Philadelphia, PA, 1989. SIAM.
19. X. Jiao, M. T. Campbell, and M. T. Heath. *occom*: an object-oriented, data-centric software integration framework for multiphysics simulations. In *ICS*, pages 358–368, 2003.
20. J. C. Jill Dahlburg and et al. Fusion Simulation Project: Integrated Simulation and Optimization of Magnetic Fusion Systems. *Journal of Fusion Energy*, 20(4):135–196, 2001. also see <http://www.isofs.info>.
21. S. Kohn, G. Kumfert, J. Painter, and C. Ribbens. Divorcing Language Dependencies from a Scientific Software Library. In *10th SIAM Conference on Parallel Processing*, Portsmouth, VA, March 12-14 2001. LLNL document UCRL-JC-140349. See also <http://www.llnl.gov/CASC/components/babel.html>.
22. J. Larson, R. Jacob, and E. Ong. The model coupling toolkit: A new fortran90 toolkit for building multiphysics parallel coupled models. *International Journal of High Performance Computing Application*, 19:277–292, 2005.
23. F. Liu and R. Bramley. CCA-LISI: On Designing a CCA Parallel Sparse Linear Solver Interface. In *Proc. 21th Int'l Parallel & Distributed Processing Symp.(IPDPS)*. ACM/IEEE Computer Society, 2007. 10 pages.
24. F. Liu, M. Sosonkina, and R. Bramley. A HPC Sparse Solver Interface for Scalable Multilevel Methods. In *High Performance Computing Symposium*, March 22-27 2009.
25. Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2003.
26. Y. Shengsheng, W. Yuanqiao, H. Liwen, and D. jian. Framework of Distributed Numerical Model Coupling System. In *DS-RT '05: Proceedings of the 9th IEEE International Symposium on Distributed Simulation and Real-Time Applications*, pages 187–194, Washington, DC, USA, 2005. IEEE Computer Society.
27. M. Sosonkina, F. Liu, and R. Bramley. Usability Levels for Sparse Linear Algebra Components. *Concurrency and Computation: Practice and Experience*, 20:1439–1454, 2008.
28. C. Szyperski. *Component Software: Beyond Object-Oriented Programming*. ACM Press, New York, 1999.
29. G. Toth, O. Volberg, and et al. A Physics-Based Software Framework for Sun-Earth Connection Modeling. In In A. T. Y. Liu, Y. Kamide, and G. Consolini (Eds), *Multiscale Coupling of Sun-Earth Processes, Proceeding of the Conference on the Sun-Earth Connection*, pages 383–397, 2004.
30. F. Zhang. *The Schur Complement and Its Application*. Numerical Methods and Algorithms. Springer, 2005.
31. S. J. Zhou. Coupling climate models with the earth system modeling framework and the common component architecture. *Concurr. Comput. : Pract. Exper.*, 18(2):203–213, 2006.